SPC560B40/50x, SPC560C40/50x
Errata sheet

32-bit MCU family built on the Power Architecture® embedded category for automotive body electronics applications

Introduction

This errata sheet describes all the known functional and electrical limitations of the SPC560B40x/50x and SPC560C40x/50x microcontroller family.

All the topics covered in this document refer to *RM0017* rev 9 and *SPC560Bx and SPC560Cx datasheet* rev 13 (see *Section A.1: Reference documents*).

The device identification can be read by software using the MIDR1 register:

- MAJOR_MASK[3:0]: 1
- MINOR_MASK[3:0]: 5

Package device marking mask identifier: BF.
Die mask ID: FB50BFQ

This errata sheet applies to SPC560B40x/50x and SPC560C40x/50x devices in accordance with *Table 1*.

Table 1. Device summary

256 Kbyte Code Flash		512 Kbyte Code Flash	
SPC560B40L5	—	SPC560B50L5	—
SPC560B40L3	SPC560C40L3	SPC560B50L3	SPC560C50L3
SPC560B40L1	SPC560C40L1	SPC560B50L1	SPC560C50L1

Contents

1 Functional problems

1.1 ERR002161: NPC: Automatic clock gating does not work for MCKO_DIV = 8

Description:

If the Nexus clock divider (NPC_PCR[MCKO_DIV]) in the Nexus Port Controller Port Configuration Register is set to 8 and the Nexus clock gating control (NPC_PCR[MCKO_GT]) is enabled, the nexus clock (MCKO) will be disabled prior to the completion of transmission of the Nexus message data.

Workaround:

Do not enable the automatic clock gating mode when the Nexus clock divider is set to 8. If Nexus clock gating is required, use a divide value of 1, 2 or 4 (set NPC_PCR[MCKO_GT]=0b1 and NPC_PCR[MCKO_DIV]=0b000, 0b001, or 0b011). However, MCKO must be kept below the maximum Nexus clock rate as defined in the device data sheet. If divide by 8 clock divider is required, then do not enable clock gating (set NPC_PCR[MCKO_GT]=0b0 and NPC_PCR[MCKO_DIV]=0b111).

1.2 ERR002656: FlexCAN: Abort request blocks the CODE field

Description:

An Abort request to a transmit Message Buffer (TxMB) can block any write operation into its CODE field. Therefore, the TxMB cannot be aborted or deactivated until it completes a valid transmission (by winning the CAN bus arbitration and transmitting the contents of the TxMB).

Workaround:

Instead of aborting the transmission use deactivation.

Note: Note that there is a chance that the deactivated TxMB can be transmitted without setting IFLAG and updating the CODE field if it is deactivated.

1.3 ERR002685: FlexCAN: Module Disable Mode functionality not described correctly

Description:

Module Disable Mode functionality is described as the FlexCAN block is directly responsible for shutting down the clocks for both CAN Protocol Interface (CPI) and Message Buffer Management (MBM) sub-modules. In fact, FlexCAN requests this action to an external logic.

Workaround:

In FlexCAN documentation chapter (see *Section A.1: Reference documents*):

Section "Modes of Operation", bullet "Module Disable Mode": Where is written:
"This low power mode is entered when the MDIS bit in the MCR Register is asserted. When disabled, the module shuts down the clocks to the CAN Protocol Interface and Message Buffer Management sub-modules.."

The correct description is:
"This low power mode is entered when the MDIS bit in the MCR Register is asserted by the CPU. When disabled, the module requests to disable the clocks to the CAN Protocol Interface and Message Buffer Management sub-modules."

Section "Modes of Operation Details", Sub-section "Module Disable Mode": Where is written:
"This low power mode is entered when the MDIS bit in the MCR Register is asserted. If the module is disabled during Freeze Mode, it shuts down the clocks to the CPI and MBM sub-modules, sets the LPM_ACK bit and negates the FRZ_ACK bit.."

The correct description is:
"This low power mode is entered when the MDIS bit in the MCR Register is asserted. If the module is disabled during Freeze Mode, it requests to disable the clocks to the CAN Protocol Interface (CPI) and Message Buffer Management (MBM) sub-modules, sets the LPM_ACK bit and negates the FRZ_ACK bit."

1.4 ERR002883: FMPLL: FMPLL_CR[UNLOCK_ONCE] wrongly set

Description:

If the FMPLL is locked and a functional reset occurs, FMPLL_CR[UNLOCK_ONCE] is automatically set even when the FMPLL has not lost lock.

Workaround:

Do not use the FMPLL_CR[UNLOCK_ONCE] when a functional reset occurs.

1.5 ERR002958: MC_RGM: Clearing a flag at RGM_DES or RGM_FES register may be prevented by a reset

Description:

Clearing a flag at RGM_DES and RGM_FES registers requires two clock cycles because of a synchronization mechanism. As a consequence if a reset occurs while clearing is on-going the reset may interrupt the clearing mechanism leaving the flag set.

Note: Note that this failed clearing has no impact on further flag clearing requests.

Workaround:

No workaround for all reset sources except SW reset.

Note: Note that in case the application requests a SW reset immediately after clearing a flag in RGM_xES the same issue may occur. To avoid this effect the application must ensure that flag clearing has completed by reading the RGM_xES register before the SW reset is requested.

1.6 ERR002970: VREG register is mirrored at consecutive addresses

Description:

The VREG Control Register (VREG_CTL) is mapped at address C3FE8080.
It is also mirrored at the following addresses:

- C3FE8080
- C3FE80A0
- C3FE80C0
- C3FE80E0

Access to any of the above addresses is considered a valid access and no transfer error is generated.

Workaround:

None

1.7 ERR002978: STANDBY exit time above specification

Description:

When boot from RAM at STANDBY exit is selected the latency between the STANDBY wake-up event and the release of the device reset is 80 us. In the data sheet this time is specified at 50 us.

Workaround:

None

1.8 ERR002981: FMPLL: Do not poll flag FMPLL_CR[PLL_FAIL]

Description:

For the case when the FMPLL is indicating loss of lock the flag FMPLL_CR[PLL_FAIL] is unpredictable.

Workaround:

To avoid reading an incorrect value of FMPLL_CR[PLL_FAIL] only read this flag inside the FMPLL Interrupt Service Routine (ISR). The ISR indicates the flag has been set correctly and at this point the flag can be cleared. Do not poll flag FMPLL_CR[PLL_FAIL] at any other point in the application software.

1.9 ERR002992: CAN Sampler: Second Frame mode not operable

Description:

CAN Sampler operation cannot be guaranteed in Second Frame mode (CANSAMPLER_CR[Mode=0]).

Workaround:

CAN sampler operates reliably in first frame mode (CANSAMPLER_CR[Mode=1]) for the following 2 configurations:

1. First frame mode (CANSAMPLER_CR[Mode=1]) selected and FIRC 'ON' in STANDBY.
2. First frame mode (CANSAMPLER_CR[Mode=1]) selected and FIRC 'OFF' in STANDBY with CAN baud rate less than 75kbps.

1.10 ERR002995: MC_ME: In RUNx mode after STOP0 exit system RAM can be accessed before it is ready

Description:

For the case when:
ME_STOP0_MC[FXOSC] is enabled
ME_STOP0_MC[FIRC] is disabled
ME_RUNx_MC[FIRC] is enabled
ME_RUNx_MC[SYSCLK] = FXOSC or FXOSC_DIV

At the transition of STOP0 to RUNx, the RUNx mode can be entered before the system RAM is ready. If the application software accesses the RAM during this time the RAM value can not be guaranteed.

Workaround:

There are two workarounds possible:

1. Do not disable the IRC if the system clock source is not disabled. The XOSC draws a lot more current than the IRC, so there should be no noticeable increase in the STOP0 mode power consumption.
2. Have the software check that the mode transition has completed via the ME_GS register before accessing the system RAM.

1.11 ERR002998: Wake up interrupt may be generated without any recessive to dominant transition on FlexCAN pad.

Description:

Wake up interrupt may be generated without any recessive to dominant transition on FlexCAN pad in case following conditions occur:

- FlexCAN is configured for communication.
- WAK_MSK as well as SLF_WAK bits of MCR register are set.
- apply this write access sequence to MDIS bit: set, then reset.

In this configuration, a wake up interrupt is wrongly generated when MDIS bit is clear.

Workaround:

Always program SLF_WAK and WAK_MSK bits to '0'.

1.12 ERR002999: MC_CGM and MC_PCU: A data storage exception is not generated on an access to MC_CGM or MC_PCU when the respective peripheral is disabled at MC_ME

Description:

If a peripheral with registers mapped to MC_CGM or MC_PCU address spaces is disabled via the MC_ME any read or write accesses to this peripheral will be ignored without producing a data storage exception.

Workaround:

For any mode other than a low-power mode do not disable any peripheral that is mapped to MC_CGM or MC_PCU.

1.13 ERR003001: MC_ME: ME_PSx registers may show '1' in a reserved bit field

Description:

Some bits in the ME_PSx registers that are defined to always return the value '0' may instead return the value '1'.

Workaround:

It is recommended as a general rule that the User should always ignore the read value of reserved bit fields.

1.14 ERR003010: ADC: conversion chain failing after ABORT chain

Description:

During a chain conversion while the ADC is in scan mode when ADC_MCR[ABORTCHAIN] is asserted the current chain will be aborted as expected. However, in the next scan the first conversion of the chain is performed twice and the last conversion of the chain is not performed.

Workaround:

When aborting a chain conversion enable ADC_MCR[ABORTCHAIN] and disable ADC_MCR[START].

ADC_MCR[START] can be enabled when the abort is complete.

1.15 ERR003012: RGM: Register RGM_FES bit PLL_FAIL is set in case of LVD2.7

Description:

When the PLL is used and a low voltage event is detected on LVD2.7 at the register RGM_FES the bit PLL_FAIL may be set.

Workaround:

None

1.16 ERR003021: LINFlex: Unexpected LIN timeout in slave mode

Description:

If the LINFlex is configured in LIN slave mode, an unexpected LIN timeout event (LINESR[OCF]) may occur during LIN Break reception.

Workaround:

It is recommended to disable this functionality during LINFlex initialization by clearing LINTCSR[IOT] and LINIER[OCIE] bits, and ignore timeout events.

1.17 ERR003028: LINFlex: BDRL/BDRM cannot be accessed as byte or half-word

Description:

LINFlex data buffers (BDRL/BDRM) cannot be accessed as byte or halfword. Accessing BDRL/BDRM in byte/half word mode will lead to incorrect data writing/reading.

Workaround:

Access BDRL/BDRM registers as word only.

1.18 ERR003032: Auto clock off feature does not work for adclksel=1

Description:

Auto clock off feature in which clock to ADC analog is automatically stopped when in pwdn mode or in IDLE mode with no conversion ongoing does not work when ADC analog clock is same as ADCDigital interface clock i.e. adclksel = 1.

This means a little higher power consumption when this configuration is used.

Workaround:

None

1.19 ERR003049: MC_RGM: External Reset not asserted if Short Reset enabled

Description:

For the case when the External Reset is enabled for a specific reset source at RGM_FBRE and a short reset is requested for the same reset source at RGM_FESS the External Reset is not asserted.

Workaround:

None

1.20 ERR003060: MC_RGM: SAFE mode exit may be possible even though condition causing the SAFE mode request has not been cleared

Description:

A SAFE mode exit should not be possible as long as any condition that caused a SAFE mode entry is still active. However, if the corresponding status flag in the RGM_FES register has been cleared, the SAFE mode exit may incorrectly occur even though the actual condition is still active.

Workaround:

Software must clear the SAFE mode request condition at the source before clearing the corresponding RGM_FES flag. This will ensure that the condition is no longer active when the RGM_FES flag is cleared and thus the SAFE mode exit can occur under the correct conditions.

1.21 ERR003064: RAM: No data abort exception generated above 0x4000BFFF

Description:

System RAM reserved space extends from 0x40000000 to 0x400FFFFF. On this device, Memory is implemented from 0x40000000 to 0x4000BFFF.

No data abort error is generated when accessing 0x4000C000-0x400FFFFF address range.

Workaround:

Use Memory Protection Unit when specific control is to be implemented for out of memory accesses.

1.22 ERR003080: ADC: CTUEN bit in ADC.MCR register cannot be reset if a BCTU channel is enabled

Description:

While any BCTU channels is enabled (CTU.EVTCFGRx.TM =1), the CTU will continuously send trigger requests to ADC. If CTUEN bit in MCR is reset while BCTU channels are

enabled, the ADC DTU trigger state may become undefined and ADC module may not service trigger request from CTU anymore.

Workaround:

Ensure ADC.MCR.CTUEN is set before enabling any CTU channels (CTU.EVTCFGRx.TM =1).
Ensure all CTU channels are disabled (CTU.EVTCFGRx.TM =0) before ADC.MCR.CTUEN is cleared.

1.23 ERR003086: MC_RGM: External reset not asserted if STANDBY0 is exited due to external reset event

Description:

If the device is in STANDBY0 mode and an external reset occurs, the MC_RGM may not assert the external reset for the duration of the reset sequence even when RGM_FBRE[BE_EXR = 0]. This incorrect behavior occurs only if the system releases the external reset before the end of reset sequence PHASE1.

Workaround:

Ensure that the system asserts the external reset for at least the maximum duration of reset sequence PHASE1.

1.24 ERR003114: FLASH: Erroneous update of the ADR register in case of multiple ECC errors

Description:

An erroneous update of the Address register (ADR) occurs whenever there is a sequence of 3 or more events affecting the ADR (ECC single or double bit errors or RWW error) and both the following conditions apply:

- The priorities are ordered in such a way that only the first event should update ADR.
- The last event although it does not update ADR sets the Read While Write Event Error (RWE) or the ECC Data Correction (EDC) in the Module Configuration Register (MCR).

For this case the ADR is wrongly updated with the address related to one of the intervening events.

Example 1 If a sequence of two double-bit ECC errors is followed by a single-bit correction without clearing the ECC Event Error flag (EER) in the MCR, then the value found in ADR after the single-bit correction event is the one related to the second double-bit error (instead of the first one, as specified)

Workaround:

Always process Flash ECC errors as soon as they are detected.

Clear MCR[RWE] at the end of each flash operation (Program, Erase, Array Integrity Check, etc...).

1.25 ERR003119: SWT: SWT interrupt does not cause STOP0 mode exit

Description:

While in STOP0 mode, if the SWT is configured to generate an interrupt and the system clock is disabled (ME_STOP0_MC[SYSCLK] = 0xF), a SWT interrupt event will not trigger an exit from STOP0 mode.

Other internal or external wakeup events (RTC, API, WKUP pins) are not affected and will trigger a STOP0 exit independent of the ME_STOP0_MC[SYSCLK] configuration.

Workaround:

If a SWT interrupt is to be used to wake the device during STOP0 mode, software may not disable the system clock (ME_STOP0_MC[SYSCLK] != 0xF).

1.26 ERR003156: Incorrect Watch-dog period value on reset exit

Description:

Watch-dog initial period may vary from 7ms to 13ms with respect to the typical 10ms value

Workaround:

Ensure to reload watch-dog and update watch-dog counter value at the beginning of application, latest 7ms after the internal reset has been released.

1.27 ERR003190: MC_ME: Main VREG not disabled during STOP0 or HALT0 mode if RUN[0..3] mode selects FXOSC to be running and target mode selects FXOSC as system clock

Description:

If STOP0 or HALT0 is configured with ME_[mode]_MC.MVRON = '0', ME_[mode]_MC.FIRCON = '0' and ME_[mode]_MC.SYSCLK = '0010/0011' the Main VREG will nevertheless remain enabled during the STOP0 mode if the previous RUN[0..3] mode is configured with ME_RUN[0..3]_MC.FXOSCON = '1'. This will result in increased current consumption of 500uA than expected.

Workaround:

Before entering STOP0 or HALT0 mode with the following configuration - ME_[mode]_MC.MVRON = '0', ME_[mode]_MC.FIRCON = '0' and ME_[mode]_MC.SYSCLK = '0010/0011' - ensure the RUN[0..3] mode switches off FXOSC - ME_RUN[0..3]_MC.FXOSCON = '0' before attempting to low power mode transition.

1.28 ERR003198: F_SOFT bit set on STANDBY exit through external reset

Description:

When device is under standby and external reset is triggered, the device exit reset and RGM_FES[F_EXR] is correctly set. RGM_FES[F_SOFT] is instead incorrectly set even if no software reset was requested by application.

Workaround:

None

1.29 ERR003200: PIT events cannot be used to trigger ADC conversion in case BCTU runs on divided system clock

Description:

If BCTU operates on divided system clock (i.e MC_CGM.CGM_SC_DC2.DIV0 NOT EQUAL 0x0), events from PIT timer cannot be used to trigger ADC conversion. In this case BCTU fails to latch the channel number to be converted. So BCTU will send the conversion request to ADC but the channel number will correspond to previous non-PIT conversion request or 0th channel in case no previous non-PIT event has occurred

Workaround:

Always write MC_CGM.CGM_SC_DC2.DIV0 to 0x0 to run BCTU at system frequency.

1.30 ERR003202: MC_ME: Invalid Configuration not flagged if PLL is on while OSC is off

Description:

PLL clock generation requires oscillator to be on. Mode configuration in which PLLON bit is "1" and OSCON bit is "0" is an invalid mode configuration. When ME_XXX_MC registers are attempted with such an invalid configuration, ME_IS.I_ICONF is not getting set which is wrong. Eventually the mode transition did not complete and system hangs.

Workaround:

Always program Oscillator to be on when PLL is required.

1.31 ERR003209: NMI pin configuration limitation in standby mode

Description:

NMI pin cannot be configured to generate Non Maskable Interrupt event to the core (WKPU_NCR[NDSS] = "00") if the following standby mode is to be used:

- NMI pin enabled for wake-up event,
- standby exit sequence boot from RAM,
- code flash module power-down on standby exit sequence.

With the following configuration the following scenario may happen:

1. System is in standby
2. NMI event is triggered on PA[1]
3. System wakeup z0 core power domain.
4. z0 core reset is released and NMI event is sampled by core on first clock-edge.
5. z0 core attempt to fetch code at 0x10 address (IVPR is not yet initialized by application) and receive an exception since flash is not available
6. z0 core enter machine check and execution is stalled.

Workaround:

If NMI is configured as a wake-up source, WKPU_NCR[NDSS] must be configured as "11". This will ensure no NMI event is triggered on the core while the system wakeup is triggered.

After standby exit, core will boot and configure its IVOR/IVPR, it may then re-configure WKPU_NCR:DSS to the appropriate configuration for enabling NMI/CI/MCP.

1.32 ERR003210: PA[1] pull-up enabled when NMI activated

Description:

When NMI is enabled (either WKPU_NCR[NREE] or WKPU_NCR[NFEE] set to '1'), PA[1] pull-up is automatically activated independently from SIUL_PCR1[WPS] and SIUL_PCR1[WPE].

This has no effect during STANDBY mode. PA[1] pull-up is then correctly configured through WKPU_WIPUER[IPUE[2]].

Workaround:

None

1.33 ERR003219: MC_CGM: System clock may stop for case when target clock source stops during clock switching transition

Description:

The clock switching is a two step process. The availability of the target clock is first verified. Then the system clock is switched to the new target clock source within two target clock cycles.

For the case when the FXOSC stops during the required two cycles, the switching process may not complete, causing the system clock to stop and prevent further clock switching. This may happen if one of the following cases occurs while the system clock source is switching to FXOSC:

- FXOSC oscillator failure
- SAFE mode request occurs, as this mode will immediately switch OFF the FXOSC (refer to ME_SAFE_MC register configuration)

Workaround:

The device is able to recover through any reset event ('functional', 'destructive', internal or external), so typically either the SWT (internal watchdog) will generate a reset or, in case it is used in the application, the external watchdog will generate an external reset. In all cases the devices will restart properly after reset.

To reduce the probability that this issue occurs in the application, disable SAFE mode transitions when the device is executing a mode transition with the FXOSC as the system clock source in the target mode.

1.34 ERR003240: PB[10] configuration during stand-by

Description:

PB[10] is a pin with several functionality, including wake-up functionality and analog functionality.

As for all wake-up pin, it must be driven either high level or low level (possibly using the internal pull- up) during standby.

In case the pin is connected to external component providing analog signal, it is important to check that this external analog signal is either lower than 0.2*VDD_HV or higher than 0.8*VDD_HV not to incur extra consumption.

Workaround:

None

1.35 ERR003247: MC_ME: A mode transition will not complete if the FlexCAN is disabled for target mode at MC_ME and is enabled at the FlexCAN peripheral

Description:

If a FlexCAN module is enabled for the current mode at MC_ME using the ME_RUN_PCx/ME_PCTLx registers and also enabled at the FlexCAN Module Configuration Register, for the case when the target mode (run or low power) disables the FlexCAN module, this transition will only complete if the FlexCAN is disabled at the FlexCAN peripheral prior to the target mode transition.

Workaround:

Before initiating the target mode change at the MC_ME the FlexCAN Module Configuration Register should be configured to set Freeze Enable, Halt and Module Disable (FLEXCAN_MCR) i.e. FLEXCAN_MCR[FRZ] = FLEXCAN_MCR[HALT] = FLEXCAN_MCR[MDIS] = 1.

1.36 ERR003407: FlexCAN: CAN Transmitter Stall in case of no Remote Frame in response to Tx packet with RTR=1

Description:

FlexCAN does not transmit an expected message when the same node detects an incoming Remote Request message asking for any remote answer.

The issue happens when two specific conditions occur:

1. The Message Buffer (MB) configured for remote answer (with code "a") is the last MB. The last MB is specified by Maximum MB field in the Module Configuration Register (MCR[MAXMB]).
2. The incoming Remote Request message does not match its ID against the last MB ID.

While an incoming Remote Request message is being received, the FlexCAN also scans the transmit (Tx) MBs to select the one with the higher priority for the next bus arbitration. It is expected that by the Intermission field it ends up with a selected candidate (winner). The coincidence of conditions (1) and (2) above creates an internal corner case that cancels the Tx winner and therefore no message will be selected for transmission in the next frame. This gives the appearance that the FlexCAN transmitter is stalled or "stops transmitting".

The problem can be detectable only if the message traffic ceases and the CAN bus enters into Idle state after the described sequence of events.

There is NO ISSUE if any of the conditions below holds:

a) The incoming message matches the remote answer MB with code "a".

b) The MB configured as remote answer with code "a" is not the last one.

c) Any MB (despite of being Tx or Rx) is reconfigured (by writing its CS field) just after the Intermission field.

d) A new incoming message sent by any external node starts just after the Intermission field.

Workaround:

Do not configure the last MB as a Remote Answer (with code "a").

1.37 ERR003442: CMU monitor: FXOSC/FIRC and FMPLL/FIRC relation

Description:

Functional CMU monitoring can only be guaranteed when the following conditions are met:

- FXOSC frequency must be greater than (FIRC / 2^RCDIV) + 0.5MHz in order to guarantee correct FXOSC monitoring
- FMPLL frequency must be greater than (FIRC / 4) + 0.5MHz in order to guarantee correct FMPLL monitoring

Workaround:

Refer to description

1.38 ERR003442: CMU monitor: FXOSC/FIRC and FMPLL/FIRC relation

Description:

Functional CMU monitoring can only be guaranteed when the following conditions are met:

- FXOSC frequency must be greater than (FIRC / 2^RCDIV) + 0.5MHz in order to guarantee correct FXOSC monitoring
- FMPLL frequency must be greater than (FIRC / 4) + 0.5MHz in order to guarantee correct FMPLL monitoring

Workaround:

Refer to description

1.39 ERR003446: CTU : The CTU (Cross Trigger Unit) CLR_FLAG in EVTCFGR register does not function as expected

Description:

If the CTU CLR_FLG is set and the CTU is idle, a PIT triggered request to the CTU does not result in the correct ADC channel number being latched. The previous ADC channel number is latched instead of the requested channel number.

Workaround:

There is no software workaround to allow the CLR_FLAG functionality to operate correctly. Do not program the CLR_FLAG bit to '1'.

1.40 ERR003449: DEBUG: Device may hang due to external or 'functional' reset while using debug handshaking mechanism

Description:

If the low-power mode debug handshake has been enabled and an external reset or a 'functional' reset occurs while the device is in a low-power mode, the device will not exit reset.

Workaround:

The NPC_PCR[LP_DBG_EN] bit must be cleared to ensure the correct reset sequence.

1.41 ERR003512: ECSM: ECSM_PFEDR displays incorrect endianness

Description:

The ECSM_PFEDR register reports ECC data using incorrect endiannes. For example, a flash location that contains the data 0xAABBCCDD would be reported as 0xDDCCBBAA at ECSM_PFEDR.

This 32-bit register contains the data associated with the faulting access of the last, properly-enabled flash ECC event. The register contains the data value taken directly from the data bus.

Workaround:

Software must correct endianness.

1.42 ERR003570: MC_ME: Possibility of Machine Check on Low-Power Mode Exit

Description:

When executing from the flash and entering a Low-Power Mode (LPM) where the flash is in low- power or power-down mode, 2-4 clock cycles exist at the beginning of the RUNx to LPM transition during which a wakeup or interrupt will generate a checkstop due to the flash not being available on RUNx mode re-entry. This will cause either a checkstop reset or machine check interrupt.

Workaround:

If the application must avoid the reset, two workarounds are suggested:

1. Configure the application to handle the machine check interrupt in RAM dealing with the problem only if it occurs
2. Configure the MCU to avoid the machine check interrupt, executing the transition into low power modes in RAM

There is no absolute requirement to work around the possibility of a checkstop reset if the application can accept the reset, and associated delays, and continue. In this event, the WKPU.WISR will not indicate the channel that triggered the wakeup though the F_CHKSTOP flag will indicate that the reset has occurred. The F_CHKSTOP flag could still

be caused by other error conditions so the startup strategy from this condition should be considered alongside any pre-existing strategy for recovering from an F_CHKSTOP condition.

1.43 ERR004146: When an ADC conversion is injected, the aborted channel is not restored under certain conditions

Description:

When triggered conversions interrupt the ADC, it is possible that the aborted conversion does not get restored to the ADC and is not converted during the chain. Vulnerable configurations are:

1. Injected chain over a normal chain
2. CTU trigger over a normal chain
3. CTU trigger over an injected chain

When any of these triggers arrive whilst the ADC is in the conversion stage of the sample and conversion, the sample is discarded and is not restored. This means that the channel data register will not show the channel as being valid and the CEOCFRx field will not indicate a pending conversion. The sample that was aborted is lost.

When the trigger arrives during the final channel in a normal or injected chain, the same failure mode can cause two ECH/JECH interrupts to be raised.

If the trigger arrives during the sampling phase of the last channel in the chain, an ECH is triggered immediately, the trigger is processed and the channel is restored and after sampling/conversion, a second ECH interrupt occurs.

In scan mode, the second ECH does not occur if the trigger arrives during the conversion phase. In one-shot mode, the trigger arriving during the conversion phase of the last channel restarts the whole conversion chain and the next ECH occurs at completion of that chain.

Workaround:

It is suggested that the application check for valid data using the CDR status bits or the CEOCFRx registers to ensure all expected channels have converted. This can be tested by running a bitwise AND and an XOR with either the JCMRx or NCMRx registers and the CEOCFRx registers during the ECH of JECH handler. Any non-zero value for (xCMRx & (xCMRx CEOCFRx)) indicates that a channel has been missed and conversion should be requested again.

Spurious ECH/JECH interrupts can be detected by checking the NSTART/JSTART flags in the ADC Module Status Registers – if the flag remains set during an ECH/JECH interrupt then another interrupt will follow after the restored channel or chain has been sampled and converted.

The spurious ECH/JECH workaround above applies to single-shot conversions. In single-shot mode, NSTART changes from 1 to 0. Therefore, the user can rely on checking the NSTART bit to confirm if a spurious ECH has occurred. However, for scan mode, the NSTART bit will remain set during normal operation, so it cannot be relied upon to check for the spurious ECH/JECH issue. Consequently, if CTU is being used in trigger mode, the conversions must be single-shot and not scan mode.

1.44 ERR004168: ADC: Abort switch aborts the ongoing injected channel as well as the upcoming normal channel

Description:

If an Injected chain (jch1,jch2,jch3) is injected over a Normal chain (nch1,nch2,nch3,nch4) the Abort switch does not behave as expected.

Expected behavior:

Correct Case (without SW Abort on jch3): Nch1- Nch2(aborted) -Jch1 - Jch2 - Jch3 - Nch2(restored) Nch3 - Nch4

Correct Case(with SW Abort on jch3): Nch1 - Nch2(aborted) -Jch1 - Jch2 - Jch3(aborted) - Nch2(restored) - Nch3 - Nch4

Observed unexpected behavior:

Fault1 (without SW abort on jch3): Nch1 - Nch2(aborted) - Jch1 - Jch2 - Jch3 - Nch3 - Nch4 (Nch2 not restored)

Fault2 (with SW abort on jch3): Nch1- Nch2 (aborted) - Jch1 - Jch2 - Jch3(aborted) - Nch4 (Nch2 not restored & Nch3 conversion skipped)

Workaround:

It is possible to detect the unexpected behavior by using the CEOCFRx register. The CEOCFRx fields will not be set for a not restored or skipped channel, which indicates this issue has occurred. The CEOCFRx fields need to be checked before the next Normal chain execution (in scan mode). The CEOCFRx fields should be read by every ECH interrupt at the end of every chain execution.

1.45 ERR004186: ADC: Do not trigger ABORT or ABORTCHAIN prior to the start of CTU triggered ADC conversions and do not trigger ABORTCHAIN prior to the start of INJECTED triggered ADC conversions.

Description:

When ADC_MCR[ABORT] or ADC_MCR[ABORTCHAIN] is set prior to the ADC receiving a CTU trigger, the next CTU triggered ADC conversion will not be performed and further CTU triggered ADC conversions will be blocked.

When ADC_MCR[ABORTCHAIN] is set prior to the ADC receiving an INJECTED trigger, the next INJECTED ADC conversion will not be performed. Following the ABORTCHAIN command the MCU behaviour does not meet the specification as ADC_ISR[JECH] is not set and ADC_MCR[ABORTCHAIN] is not cleared.

Workaround:

Do not program ADC_MCR[ABORT] or ADC_MCR[ABORTCHAIN] before the start of ADC conversions.

The case when CTU triggered ADC conversions are blocked should be avoided however it is possible to reactivate CTU conversions by clearing and setting ADC_MCR[CTUEN].

1.46 ERR005569: ADC: The channel sequence order will be corrupted when a new normal conversion chain is started prior to completion of a pending normal conversion chain

Description:

If One shot mode is configured in the Main Configuration Register (MCR[MODE] = 0) the chained channels are automatically enabled in the Normal Conversion Mask Register 0 (NCMR0). If the programmer initiates a new chain normal conversion, by setting MCR[NSTART] = 0x1, before the previous chain conversion finishes, the new chained normal conversion will not follow the requested sequence of converted channels.

For example, if a chained normal conversion sequence includes three channels in following sequence: channel0, channel1 and channel2, the conversion sequence is started by MCR[NSTART] = 0x1. The software re-starts the next conversion sequence when MCR[NSTART] is set to 0x1 just before the current conversion sequence finishes.

The conversion sequence should be: channel0, channel1, channel2, channel0, channel1, channel2.

However, the conversion sequence observed will be: channel0, channel1, channel2, channel1, channel1, channel2. Channel0 is replaced by channel1 in the second chain conversion and channel1 is converted twice.

Workaround:

Ensure a new conversion sequence is not started when a current conversion is ongoing. This can be ensured by issuing the new conversion setting MCR[NSTART] only when MSR[NSTART] = 0.

Note: MSR[NSTART] indicates the present status of conversion. MSR[NSTART] = 1 means that a conversion is ongoing and MSR[NSTART] = 0 means that the previous conversion is finished.

1.47 ERR006082: LINFlexD : LINS bits in LIN Status Register(LINSR) are not usable in UART mode.

Description:

When the LINFlexD module is used in the Universal Asynchronous Receiver/Transmitter (UART) mode, the LIN state bits (LINS3:0]) in LIN Status Register (LINSR) always indicate the value zero. Therefore, these bits cannot be used to monitor the UART state.

Workaround:

LINS bits should be used only in LIN mode.

1.48 ERR006620: FLASH: ECC error reporting is disabled for Address Pipelining Control (APC) field greater than Read Wait-State Control (RWSC) field.

Description:

The reference manual states the following at the Platform flash memory controller Access pipelining functional description.

“The platform flash memory controller does not support access pipelining since this capability is not supported by the flash memory array. As a result, the APC (Address Pipelining Control) field should typically be the same value as the RWSC (Read Wait-State Control) field for best performance, that is, BKn_APC = BKn_RWSC. It cannot be less than the RWSC.”

The reference manual advises that the user must not configure APC to be less than RWSC and typically APC should equal RWSC. However the documentation does not prohibit the configuration of APC greater than RWSC and for this configuration ECC error reporting will be disabled. Flash ECC error reporting will only be enabled for APC = RWSC.

For the case when flash ECC is disabled and data is read from a corrupt location the data will be transferred via the system bus however a bus error will not be asserted and neither a core exception nor an ECSM interrupt will be triggered. For the case of a single-bit ECC error the data will be corrected but for a double-bit error the data will be corrupted.

Note: Both CFlash & DFlash are affected by this issue.

For single-bit and double-bit Flash errors neither a core exception nor an ECSM interrupt will be triggered unless APC=RWSC.

The Flash Array Integrity Check feature is not affected by this issue and will successfully detect an ECC error for all configurations of APC >= RWSC.

For the APC > RWSC configuration other than flash ECC error reporting there will be no other unpredictable behaviour from the flash.

The write wait-state control setting at PFCRx[BKn_WWSC] has no effect on the flash.It is recommended to set WWSC = RWSC = APC.

Workaround:

PFCRx[BKy_APC] must equal PFCRx PFCRx[BKy_RWSC]. See datasheet for correct setting of RWSC.

1.49 ERR006726: NPC: MCKO clock may be gated one clock period early when MCKO frequency is programmed as SYS_CLK/8.and gating is enabled

Description:

The Nexus auxiliary message clock (MCKO) may be gated one clock period early when the MCKO frequency is programmed as SYS_CLK/8 in the Nexus Port Controller Port Configuration Register (NPC_PCR[MCKO_DIV]=111) and the MCKO gating function is enabled (NPC_PCR[MCKO_GT]=1). In this case, the last MCKO received by the tool prior to the gating will correspond to the END_MESSAGE state. The tool will not receive an MCKO to indicate the transition to the IDLE state, even though the NPC will transition to the IDLE state internally. Upon re-enabling of MCKO, the first MCKO edge will drive the Message Start/End Output (MSEO=11) and move the tool's state to IDLE.

Workaround:

Expect to receive the MCKO edge corresponding to the IDLE state upon re-enabling of MCKO after MCKO has been gated.

1.50 ERR006976: MC_ME: SAFE mode not entered immediately on hardware-triggered SAFE mode request during STOP0 mode

Description:

If a SAFE mode request is generated by the Reset Generation Module (MC_RGM) while the chip is in STOP0 mode, the chip does not immediately enter SAFE mode if STOP0 is configured as follows in the STOP0 Mode Configuration register (ME_STOP0_MC):

- the system clock is disabled (ME_STOP0_MC[SYSCLK] = 0b1111)
- the internal RC oscillator is enabled (ME_STOP0_MC[IRCON] = 0b1)

In this case, the chip will remain in STOP0 mode until an interrupt request or wakeup event occurs, causing the chip to return to its previous RUNx mode, after which the still pending SAFE mode request will cause the chip to enter SAFE mode.

Workaround:

There are two possibilities.

1. Configure the internal RC oscillator to be disabled during STOP0 mode (ME_STOP0_MC[IRCON] = 0b0) if the device supports it.
2. Prior to entering STOP0 mode, configure all hardware-triggered SAFE mode requests that need to cause an immediate transition from STOP0 to SAFE mode to be interrupt requests. This is done in the MC_RGM's ‘Functional' Event Alternate Request register (RGM_FEAR).

1.51 ERR007322: FlexCAN: Bus Off Interrupt bit is erroneously asserted when soft reset is performed while FlexCAN is in Bus Off state

Description:

Under normal operation, when FlexCAN enters in Bus Off state, a Bus Off Interrupt is issued to the CPU if the Bus Off Mask bit (CTRL[BOFF_MSK]) in the Control Register is set. In consequence, the CPU services the interrupt and clears the ESR[BOFF_INT] flag in the Error and Status Register to turn off the Bus Off Interrupt.

In continuation, if the CPU performs a soft reset after servicing the bus off interrupt request, by either requesting a global soft reset or by asserting the MCR[SOFT_RST] bit in the Module Configuration Register, once MCR[SOFT_RST] bit transitions from 1 to 0 to acknowledge the soft reset completion, the ESR[BOFF_INT] flag (and therefore the Bus Off Interrupt) is re-asserted.

The defect under consideration is the erroneous value of Bus Off flag after soft reset under the scenario described in the previous paragraph.

The Fault Confinement State (ESR[FLT_CONF] bit field in the Error and Status Register) changes from 0b11 to 0b00 by the soft reset, but gets back to 0b11 again for a short period, resuming after certain time to the expected Error Active state (0b00). However, this late correct state does not reflect the correct ESR[BOFF_INT] flag which stays in a wrong value and in consequence may trigger a new interrupt service.

Workaround:

To prevent the occurrence of the erroneous Bus Off flag (and eventual Bus Off Interrupt) the following soft reset procedure must be used:

1. Clear CTRL[BOFF_MSK] bit in the Control Register (optional step in case the Bus Off Interrupt is enabled).
2. Set MCR[SOFT_RST] bit in the Module Configuration Register.
3. Poll MCR[SOFT_RST] bit in the Module Configuration Register until this bit is cleared.
4. Wait for 4 peripheral clocks.
5. Poll ESR[FLTCONF] bit in the Error and Status Register until this field is equal to 0b00.
6. Write "1" to clear the ESR[BOFF_INT] bit in the Error and Status Register.
7. Set CTRL[BOFF_MSK] bit in the Control Register (optional step in case the Bus Off Interrupt is enabled).

1.52 ERR007332 I2C:IBSR[RXAK] bit sets even without data/address transmission

Description:

Receive acknowledge bit (IBSR[RXAK]) in the Inter Integrated Circuit (I2C) status register is an indication that the acknowledgement signal has been received after the transmission of 8 data bits on the bus. But IBSR[RXAK] bit is getting set even without any data/address transmission.

Workaround:

Software should read IBSR[RXAK] bit only after transfer is complete i.e. IBSR[TC] bit is set because it is valid only after that.

1.53 ERR007394: MC_ME: Incorrect mode may be entered on low-power mode exit.

Description:

For the case when the Mode Entry (MC_ME) module is transitioning from a run mode (RUN0/1/2/3) to a low power mode (HALT/STOP/STANDBY*) if a wake-up or interrupt is detected one clock cycle after the second write to the Mode Control (ME_MCTL) register, the MC_ME will exit to the mode previous to the run mode that initiated the low power mode transition.

Example correct operation DRUN->RUN1-> RUN3->STOP->RUN3 Example failing operation DRUN->RUN1-> RUN3->STOP->RUN1

*Note: *STANDBY mode is not available on all MPC56xx microcontrollers*

Workaround:

To ensure the application software returns to the run mode (RUN0/1/2/3) prior to the low power mode (HALT/STOP/STANDBY*) it is required that the RUNx mode prior to the low power mode is entered twice.

The following example code shows RUN3 mode entry prior to a low power mode transition.

ME.MCTL.R = 0x70005AF0;/* Enter RUN3 Mode & Key */ ME.MCTL.R = 0x7000A50F;/* Enter RUN3 Mode & Inverted Key */|while (ME.GS.B.S_MTRANS) {} /* Wait for RUN3 mode transition to complete */ ME.MCTL.R = 0x70005AF0;/* Enter RUN3 Mode & Key */
ME.MCTL.R = 0x7000A50F;/* Enter RUN3 Mode & Inverted Key */
while (ME.GS.B.S_MTRANS) {} /* Wait for RUN3 mode transition to complete */
/* Now that run mode has been entered twice can enter low power mode */
/* (HALT/STOP/STANDBY*) when desired.*/

1.54 ERR007688: RTC: An API interrupt may be triggered prematurely after programming the API timeout value

Description:

When the API is enabled (RTCC[APIEN]), the API interrupt flag is enabled (RTCC[APIIE]) and the API timeout value (RTCC[APIVAL]) is programmed the next API interrupt may be triggered before the programmed API timeout value. Successive API Interrupts will be triggered at the correct time interval.

Workaround:

The user must not use the first API interrupt for critical timing tasks.

1.55 ERR007938: ADC: Possibility of missing CTU conversions

Description:

The CTU prioritizes and schedules trigger sources so that the ADC will receive only one CTU trigger at a time. However, whilst a Normal or Injected ADC conversion is ongoing as the ADC moves state from IDLE-to-SAMPLE , SAMPLE-to-WAIT, WAIT-to-SAMPLE and WAIT-to-IDLE there are 2 clock cycles at the state transition that a CTU trigger may be missed by the ADC.

Workaround:

To ensure all CTU triggers are received at the ADC Normal and Injected modes must be disabled.

1.56 ERR007953: ME: All peripherals that will be disabled in the target mode must have their interrupt flags cleared prior to target mode entry

Description:

Before entering the target mode, software must ensure that all interrupt flags are cleared for those peripheral that are programmed to be disabled in the target mode. A pending interrupt from these peripherals at target mode entry will block the mode transition or possibly lead to unspecified behaviour.

Workaround:

For those peripherals that are to be disabled in the target mode the user has 2 options:

1. Mask those peripheral interrupts and clear the peripheral interrupt flags prior to the target mode request.
2. Through the target mode request ensure that all those peripheral interrupts can be serviced by the core.

1.57 ERR008227: CGM & ME: The peripheral set clock must be active during a peripheral clock enable or disable request

Description:

An individual peripheral clock can be enabled or disabled for a target mode via the Mode Entry Peripheral Control register (ME_PCTL) and the Mode Entry RUN/Low Power Peripheral Configuration register (ME_RUN_PC & ME_LP_PC). For this process to complete the user must ensure that the peripheral set clock relative to the specific peripheral is enabled for the duration of the current-mode-to-target-mode transition. The peripheral set clock is configured at the Clock Generation Module System Clock Divider Configuration Register (CGM_SC_DC).

A caveat for FlexCAN is for the case when the FXOSC is selected for the CAN Engine Clock Source (FLEXCAN_CTRL[CLK_SRC]). In this instance to enable or disable the FlexCAN peripheral clock the user must ensure FXOSC is enabled through the target mode transition i.e. FXOSC must be enabled for the target mode.

Workaround:

To enable a peripheral clock:

1. Enable the peripheral set clock at CGM_SC_DC.
2. Enable the peripheral clock for the target mode at ME_PCTL & ME_RUN_PC/ ME_LP_PC.N
3. Note steps 1 & 2 are interchangeable.
4. Transition to the target mode to enable the peripheral clock.

To disable a peripheral clock:

1. Disable the peripheral clock for the target mode at ME_PCTL & ME_RUN_PC/ ME_LP_PC.
2. Transition to the target mode to disable the peripheral clock.
3. Optionally disable peripheral set clock at CGM_SC_DC. Note to check other peripherals in this peripheral set are not required.

Appendix A Further information

A.1 Reference documents

- *SPC560B40x, SPC560B50x, SPC560C40x, SPC560C50x 32-bit MCU family built on the embedded Power Architecture®* (RM0017, Doc ID 14629).
- *32-bit MCU family built on the Power Architecture® for automotive body electronics applications* (SPC560B40x, SPC560B50x, SPC560C40x, SPC560C50x datasheet, Doc ID 14619).

A.2 Acronyms

Table 2. Acronyms

Acronym	Name
ADC	Analog-to-digital converter
BAM	Boot assist mode
ECC	Error correction code
DSDR	Decode signals delay register
FIFO	First in first out
ISR	Interrupt service routine
LVD	Low voltage detector
MB	Message buffer
MPU	Memory protection unit
RCHW	Reset configuration half-word
RXGMASK	RX global mask
RXIMR	RX individual mask register
TDO	Test data output

Revision history

Table 3. Document revision history

Date	Revision	Changes
07-Jul-2015	1	Initial release.